Analysis of Non-Heuristic Planning Solution Searches and Domain Independent Heuristics

There were 3 Air Cargo Problems given for analysis. The optimal paths (sequence of actions) for these three problems are as below:

Optimal Path Solution for the Problems

Description	air_cargo_p1	air_cargo_p2	air_cargo_p3
Optimal Plan Length	6	9	12
Optimal Action Sequence	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	Load(C1, P1, SFO) Load(C2, P2, JFK) Load(C3, P3, ATL) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Fly(P3, ATL, SFO) Unload(C3, P3, SFO)	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Unload(C3, P1, JFK) Fly(P2, ORD, SFO) Unload(C2, P2, SFO) Unload(C4, P2, SFO)

Search Algorithms Evaluated

The following 3 Non-Heuristic Search Functions were evaluated for the 3 Air Cargo Problems,

1. breadth_first_search
2. depth_first_graph_search
3. uniform_cost_search

and the following 2 Domain Independent Heuristics were used with A* Search.

4. astar_search h_ignore_preconditions
5. astar_search h_pg_levelsum

The following 4 metrics were recorded for the 5 different planning algorithms.

1. Optimality in terms of Path Length
2. Time Taken (in Seconds)
3. Number of Node Expansions
4. Memory (number of New Nodes created)

Legend used in the Metric Tables below.

Worst

Best

Metric : Optimality (Path Length)

Problem/ Planning Algorithm	air_cargo_p1	air_cargo_p2	air_cargo_p3
breadth_first_search	6 : optimal (not optimal until all actions have the same cost.)	9 : optimal	12 : optimal
depth_first_graph_search	20 : Not optimal	619 : Worse	658 : Worst
uniform_cost_search	6: optimal	9: optimal	12 : optimal
astar_search h_ignore_preconditions	6: optimal	9 : optimal	12 : optimal
astar_search h_pg_levelsum	6 : optimal	9 : optimal	12 : optimal

Metric : Time Taken (Seconds)

Problem/ Planning Algorithm	air_cargo_p1	air_cargo_p2	air_cargo_p3
breadth_first_search	0.0526	17.634	228.769
depth_first_graph_search	0.0257	4.1965	386.324
uniform_cost_search	0.061	17.47	118.18
astar_search h_ignore_preconditions	0.055	5.82	23.18
astar_search h_pg_levelsum	0.56	48.42	337.413

Metric : Node {E}xpansions and {N}ew Nodes Created (Memory Consumed)

Problem/ Planning Algorithm	air_cargo_p1	air_cargo_p2	air_cargo_p3
breadth_first_search	E: 43, N: 180	E: 3343, N: 30509	E:19342, N :174325
depth_first_graph_search	E:21, N: 84	E:624, N: 5602	E:21958, N:205162
uniform_cost_search	E:55, N:224	E:4853, N:44041	E:26789, N:242081
astar_search h_ignore_preconditions	E:41, N:170	E:1450, N:13303	E:5321, N:47457

Problem/ Planning Algorithm	air_cargo_p1	air_cargo_p2	air_cargo_p3
astar_search h_pg_levelsum	E:11, N 59	E:86, N:841	E:399, N:3667

Analysis of Metrics

Planning is foremost an exercise in controlling combinatorial explosion. If there are p primitive propositions in a domain, then there are 2^p states. For complex domains, p can grow quite large. Since planning is exponentially hard, no algorithm will be efficient for all problems, but many practical problems can be solved with the heuristic methods defined in AIMA 3E Chapter 11. Effective heuristics can be derived by making a subgoal independence assumption and by various relaxations of the planning problem. Structures such as planning graphs are useful as a source of heuristics.

1. The A* search always returns an optimal solution first, provided that :
    1. the branching factor is finite,
    2. arc costs are bounded above 0.
    3. the heuristic function h(n) is admissible. In other words the heuristic function is non negative and an underestimate of the cost of shortest path from n to the goal node.

2. In the experiments above we used 2 out of the 3 heuristics defined in AIMA 3E Chapter 11.
    1. h_ignore_preconditions : The simplest idea is to relax the problem by *removing all preconditions* from the actions. Then every action will always be applicable. This almost implies that the number of steps required to solve a conjunction of goals is the number of unsatisfied goals.
    2. h_pg_levelsum : The sum of the level costs of the individual goals (admissible if goals independent). We used the PlanningGraph to compute this hueristic.

3. The ignore_preconditions heuristic with A* performs the best in terms of providing an optimal solution with lowest Time Taken across all 3 Air Cargo Problems.

4. The levelsum heuristic with A* on the other hand provides an optimal solution with lowest memory consumption (number of node expansions and new node creations). The time consumed by this approach however is larger than ignore_preconditions. This is because ignore_preconditions is a very simple/low cost heuristic to compute whereas the PlanningGraph based levelsum heuristic approach would need to update the planning graph as the search progresses and would need to compute the levelsum on the planning graph.

5. DFS (Depth First Search) did not produce an optimal solution. Consider the scenario that there is more than one goal node, and the search expanded the left subtree from the root first and it so happens that there is a solution at a very deep level of this left subtree, at the same time if the right subtree of the root had a solution near the root then DFS would not find it. DFS is also not complete, if it is exploring an infinite path then even though the goal is at a finite depth on another path, it would keep going down the infinite path and not terminate. DFS however has very modest memory requirements, it needs to store only the path from the root to the leaf node, beside the siblings of each node on the path, whereas BFS needs to store all the explored nodes in memory.

6. BFS (Breadth First Graph Search) in general can find a solution if there a finite path to a goal, even on an infinite graph. Breadth-first search is not optimal until all actions have the same cost. In the experiments above, BFS did find the optimal path because of the same arc cost assumption.

7. Uniform Cost search expands the node with the lowest path cost. This algorithm is complete and is always optimal (if the cost of each step exceeds some small positive integer). But in terms of Time Taken and Memory requirements it only showed an incremental improvement over BFS in the

experiments above. UCS is guided by path cost rather than path length so it is hard to determine its complexity in terms of b and d, so if we consider C to be the cost of the optimal solution, and every action costs at least e, then the algorithm worst case space and time complexity is O(b^C/e).